Assignment 5

1. Many algorithms in computational linear algebra are analytical rather than algebraic. An *algebraic* algorithm is one that produces the exact answer in a finite number of steps if the operations are performed in exact arithmetic. An *analytical* algorithm is one that gives a sequence of approximate solutions, S_n, that converges to the exact solution S as $n \to \infty$. Analytical algorithms never give the exact answer (in exact arithmetic[1]), but they may be faster and even more accurate (in floating point arithmetic) than algebraic algorithms. This exercise explores infinite matrix sums. These may be matrix versions of Taylor series. The infinite matrix sum is

$$S = \sum_{k=0}^{\infty} A_k \ .$$

The *partial sums* are a sequence of approximations to S. If the sum converges rapidly then S_n is close to S without n being very large:

$$S_n = \sum_{k=0}^{n} A_k \ .$$

The convergence rate is determined by the "tail sums"

$$S - S_n = \sum_{k>n} A_k \ .$$

The triangle inequality gives bounds on the matrix norm of the tail sum: inequality

$$\|S - S_n\| \leq \sum_{k>n} \|A_k\| \ . \tag{1}$$

There is an inequality of this form for any matrix norm that satisfies the triangle inequality.

The terms in the sum on the right of (1) are positive (technically, non-negative because $\|A_k\| = 0$ is possible) numbers. Any understanding of the size of a sum of positive numbers lead to an understanding of the tail sum on the left. The following simple tricks handle many cases:

[1]This can be misunderstood to imply that analytical algorithms are less accurate than exact ones on the computer. That is true sometimes, but not always. In floating point, the rounding errors in a supposedly exact calculation can lead to errors larger than those from an inexact analytical approximation.

- If $0 \leq r < 1$, then

$$\sum_{k=0}^{\infty} r^k = \frac{1}{1-r} \;, \qquad \sum_{k>n} r^k = r^{n+1} \sum_{k \geq 0} r^k = \frac{r^{n+1}}{1-r} \;.$$

  Therefore, if $|a_k| \leq Cr^k$, then

$$\left| \sum_{k>n} a_k \right| \leq C \, \frac{r^{n+1}}{1-r} \;.$$

  Notice that if r is only a little less than 1, the denominator is large and the numerator goes to zero more slowly as $n \to \infty$. Geometric series converge more slowly when the ratio is close to 1.

- For any real number r, the Taylor series for e^r converges

$$e^r = \sum_{k=0}^{\infty} \frac{1}{k!} r^k \;.$$

  If r is large, the terms are growing for small k. For example, if $r = 10$ the first terms are

$$a_0 = 1 \;, \quad a_1 = 10 \;, \quad a_2 = 50 \;, \quad a_3 \approx 167 \;, \cdots \;.$$

  However, some algebra shows that $a_{k+1} < \frac{1}{2} a_k$ if $k > 2r$, so the series eventually converges like a geometric series. The series is "bad" (not an efficient way to calculate e^r), but it "works" (gives an accurate approximation, given enough computing time) for any $r > 0$. If r is a large negative number, such as $r = -10$, not only is the sum inefficient (it takes many terms for an accurate approximation), but there is so much cancellation that the answer in floating point will be inaccurate not matter how many terms you use.

(a) The "geometric series" for a matrix A would be

$$S = I + A + A^2 + \cdots \;.$$

This matrix sum is called a *Neumann series* (first syllable rhymes with "toy", second syllable is like the first syllable of "tonic"). Show that the series converges if $\|A\| < 1$. Show that the sum is

$$S = (I - A)^{-1} \;.$$

Hint. Calculate $(I - A)S$, with S written as the infinite series.

(b) Write a version of the Neumann series for $(A_0 + \Delta A)^{-1}$ that uses A_0^{-1} and ΔA and powers and products and sums of these. It should converge if A_0 is invertible and A_0 is small enough.

(c) The approximate inverse, with the "first order correction" is

$$(A_0 + \Delta A)^{-1} \approx A_0^{-1} - A_0^{-1} \Delta A A_0^{-1} \ . \tag{2}$$

The error in this approximation is

$$R = (A_0 + \Delta A)^{-1} - \left[A_0^{-1} - A_0^{-1} \Delta A A_0^{-1} \right] \ .$$

Show that

$$\|R\| \leq \frac{\left\| A_0^{-1} \right\|^3 \|\Delta A\|^2}{1 - \left\| A_0^{-1} \right\| \|\Delta A\|} \ . \tag{3}$$

Hint. Use 1b to write a series for R.
Explanation. The approximation (2) is supposed to be accurate when ΔA is small relative to A_0^{-1}, using matrix norms to measure size. In that case, the denominator in (3) is essentially 1. The correction term in (2) is linear in ΔA while the error bound has the square $\|\Delta A\|^2$. This makes R much smaller than the correction term when ΔA is small. The perturbation formula (2) is a different way to get the derivative formula from Assignment 4.

(d) The *rank* of a matrix[2] is the dimension of the vector space spanned by the columns. Show that if A has rank 1, then there are column vectors u and v with $A = uv^T$.

(e) Write the Neumann series representation for

$$\left(A_0 + uv^T \right)^{-1} \ .$$

Show that all the terms after the first two contain powers of the number $v^T A_0^{-1} u$, so the matrix geometric series reduces to a few terms plus a numerical geometric series whose sum is known. Use this to find an explicit formula for the inverse. Show that $A_0 + uv^T$ is invertible if and only if $v^T A_0 u \neq -1$ (be careful with logic: the fact that a formula doesn't work doesn't mean the object doesn't exist). This is called the *Sherman Morrison* formula. This derivation is correct if the Neumann series converges but you can check that the formula is true even if the series does not converge.

(f) Consider the *matrix exponential* series

$$S = e^A = \sum_{k=0}^{\infty} \frac{1}{k!} A^k \ . \tag{4}$$

Show that the sum converges for any square matrix A. Show that the function $S(t) = e^{tA}$ satisfies the matrix differential equation

$$\frac{d}{dt} S(t) = AS(t) \ , \quad S(0) = I \ .$$

[2]This is the *column rank*. A theorem of linear algebra says that the row rank (the dimension of the space spanned by the rows) is equal to the column rank. Thus, it makes sense to call either one the *rank*.

The matrix function $S(t)$ is sometimes called the *fundamental solution* of the differential equation.

2. Absolute values are *multiplicative* in that $|ab| = |a|\,|b|$. Matrix norms are *submultiplicative* in that it is possible that $\|AB\| < \|A\|\,\|B\|$. Show that strict inequality can happen in the matrix case. Find a matrix and a vector norm $\|\cdot\|$ so that the corresponding matrix norms are $\|A\| = 2$ and $\|A^2\| = \frac{1}{2}$.

3. This exercise explores the use of the SVD for some ill conditioned linear algebra problems. The matrix involved is a simplification of matrices that arise in real applications.

   The matrix computes the influence of sources on an outer ellipse on receivers on an inner ellipse. The ellipses have the form

$$x^2 + a^2 y^2 = r^2 \ .$$

   We take $a > 1$, so the ellipse is a circle stretched in the x direction and squashed in the y direction. We take $r = r_2$ for the outer ellipse and $r = r_1 < r_2$ for the inner one. For any angle θ and r, there is a point on the ellipse at angle θ defined by

$$x = s\cos\theta \ , \quad y = s\sin(\theta) \ .$$

   This point is on the ellipse if[3]

$$s(\theta) = \frac{r}{\sqrt{(a-1)\sin^2(\theta) + 1}} \ . \tag{5}$$

   We put m *emitter* points on the outer ellipse with uniformly spaced θ values between 0 and 2π.[4] We put n *receiver* points on the inner ellipse, also with uniformly spaced θ. The influence of emitter point with θ_j on receiver point with $\theta = k$ is

$$a_{jk} = \frac{1}{r_{jk}}$$

   Here, r_{jk} is the distance between emitter point at angle θ_j and receiver point with angle θ_k. If the source emission at point k on the outer ellipse is u_k, then the total received intensity at receiver point with θ_j is

$$v_j = \sum_{k=1}^{m} a_{jk} u_k \ .$$

[3] Reality check: s is constant if $a = 1$ (a circle), s has a maximum when $\theta = 0$ (x axis), s has a minimum when $\sin(\theta) = \pm 1$ with value $s_{\min} = r/\sqrt{a}$ and the point $x = 0$, $y = \pm s_{\min}$ is on the ellipse.

[4] This is not a good point distribution if a is large, but it simplifies this problem.

(a) Write a module to compute the $n \times m$ matrix A with $Au = v$. Compute the SVD of A (using integrated `numpy` linear algebra routines) and find the smallest d so that there is a rank d approximation B to A with relative approximation error at most ϵ:

$$\|B - A\|_2 \leq \epsilon \|A\|_2 \ .$$

Use this SVD to factor B as an $n \times k$ matrix X and a $k \times m$ matrix Y, which means $XY = B$. Take n and m at least 1000 and experiment with $n < m$, $n = m$, and $n > m$, and with various values of r_1/r_2. Comment on the behavior of k: what makes k bigger or smaller.

(b) Consider an outer circle of points $(R\cos(\phi), R\sin(\phi)$ with $R > r_2$. Suppose the point on this outer circle illuminates the outer ellipse with intensity

$$u_k(\phi) = \frac{1}{r_k(\phi)} \ .$$

here $r_k(\phi)$ is the distance between the outer ellipse point at θ_k and a source at $(R\cos(\phi), R\sin(\phi)$. Take $R = r_2 + \delta$ (δ small) so that the ϕ circle comes close to the outer ellipse. Make a movie of the vector $v(\phi) = Au(\phi)$ using color to visualize the values of u and putting the points on the inner ellipse in 2D. Try to visualize the outer ellipse values $u(\phi)$ and the location of the source point $R\cos(\phi), R\sin(\phi)$.

(c) Make plots to illustrate the SVD of A. This should involve

- Plots of σ_k as a function of k. These probably should be semi-log plots because the values of σ_k span many orders of magnitude.
- Plots of the left and right singular vectors (left and right singular vectors are similar in this example but not the same). See whether the behavior of the larger k singular vectors explains the small values of the corresponding σ_k.

If you automate the process of making these plots, you will be able to quickly experiment with different r_1, r_2, a combinations and comment on trends. Please include such automated code with what you upload for this assignment.

(d) Repeat part (b) but with a low rank approximation to A. Put the matrix B in factored form $B = XY^T$, where X and Y have k rows. Apply B using $Bu = XY^Tu = X(Y^Tu)$ (why?). Do experiments in which the low rank approximation is accurate but the low rank approximation is much faster to apply. *Hint.* Python graphics routines are so slow that the visualization may take as long as the computation. You can observe the computation speed by doing all the computations (finding all the vectors $v(\phi)$) before starting the visualization. The sample visualization code posted does that.

(e) Consider the problem of finding u so that $Au = v$ and the values of v are $v_j = 1$ if $0 \leq \theta_j \leq \frac{\pi}{2}$ and $v_j = 0$ otherwise. Use the computed SVD to solve the linear algebra problems. Visualizing the vectors u is a better way to understand the results than printing values, particularly with n or m is in the thousands. Explore as time permits:

- For the square matrix case ($n = m$), see how large n can be before A gets so nearly singular that the linear system is essentially unsolvable on the computer even though the mathematical A is never singular (take my word for the last part).
- When $m < n$, find the least squares solution. How does this differ from the "exactly determined" ($m = n$) case for small and large n?
- When $m > n$, the minimum norm solution.
- Explore the effect of regularization, which means adding $\delta \|u\|^2$ to the objective function.